MAIL STOP APPEAL BRIEF-PATENTS
PATENTS
0518-1092-1

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

In re application of

Jean-Francois BILLIARD et al.	Conf. No. 4049
Serial No. 10/562,821	Appeal No. ___________
Filed December 29, 2005	Group 2179

METHOD AND DEVICE FOR GRAPHICAL INTERFACING

APPEAL BRIEF

MAY IT PLEASE YOUR HONORS:

Applicants respectfully appeal the Final Rejection of claims 13, 15-21, and 24-31 as unpatentable under 35 U.S.C. 103(a) as set forth in the Official Action mailed February 20, 2008.

(i) **Real Party in Interest**

The real party in interest in this appeal is the assignee, Amadeus S.A.S of Biot, FRANCE.

(ii) **Related Appeals and Interferences**

None.

(iii) **Status of Claims**

Claims 13, 15-21, and 24-31 are pending.

Claims 1-12, 14, 22-23 are canceled.

The final rejection of claims 13, 15-21, and 24-31 is being appealed.

(iv) <u>Status of Amendments</u>

There are no outstanding amendments. The claims have not been amended since the amendment of November 20, 2007.

Claims 13, 15-21, and 24-31 were rejected by the Official Action mailed February 20, 2008 (the "Official Action").

(v) Summary of Claimed Subject Matter

The invention is directed to a method and device for a graphical user interface with a computer system (specification page 1, lines 15-16) wherein a client terminal interfaces with a server over a network (specification page 2, lines 3-10).

Figure 2 of the application schematically shows the process of a user request when the invention is carried out. An XML or other structured formatted message (Figure 2, element 1) representing a user action, or "event", is issued from the client and destined for the server (specification page 7, lines 19-21). The message is received and processed (Figure 2, element 2) by the server to generate instructions (Figure 2, element 3) to be interpreted by the client (specification page 7, lines 29-34). The instructions are then forwarded, in the form of a XML document, as a message back to the client (Figure 2, element 4) (specification page 7 line 32 to page 8 line 6). These instructions are then interpreted and displayed (Figure 2, element 5) by the client (specification page 8, lines 9-21; page 9, lines 1-2). Client-side interaction with the user is then possible (6) (specification page 9, line 3-6).

Claims 13, 21, and 31 are independent.

Independent Claim 13

The invention, as recited by claim 13, is a method for graphic interfacing between a user and a computer system (specification page 3, lines 13-14) in which the following operations are performed:

- inputting a user request at a client terminal (specification page 3, line 15),

- transmitting the user request from the client terminal to a server part for processing by the server part (specification page 3, line 16; specification page 7, lines 19-21; Figure 2, element 1)

- receiving a response to the request at the client terminal, the response generated by the server part (specification page 3, line 21 to page 4, line 5; Figure 2, element 4), and

- displaying the response at the client terminal (specification page 3, lines 28-29; specification page 7, lines 3-6; specification page 8, lines 19-21; Figure 2, element 5).

Claim 13 recites that the response comprises instruction data and display data to be displayed (specification page 3, lines 21-22; specification page 10, lines 6-8; Figure 3, "Server Response").

Claim 13 further recites, at the client terminal, the instruction data are interpreted and forwarded to an association engine (specification page 10, lines 10-17; Figure 3, "Association

Engine"), a visualization model is created by the association engine according to the interpreted instruction data through the association of construction elements retrieved from storage at the client terminal (specification page 10, lines 12-17), display data of the response are merged with the visualization model and displayed as a merged result (specification page 11, lines 12-15; Figure 3, "Merging Means"), and logical rules are applied to the visualization model by a rules engine (specification page 11, lines 20-24; Figure 3, "Rules Engine"), providing event-operated interface controls in the visualization model and script code to manage the event-operated interface controls at the level of the client terminal (specification page 11, lines 24-29).

Claim 13 recites that the construction elements include a descriptive interface of visualization model objects, a presentation layer, and the logical rules (specification page 10, lines 17-22; Figure 3, "Descriptive Interface", "Presentation Layer", "Logical Rules"; Figure 5, "Descriptive Interface", "Presentation Layer", "Logical Rules").

Independent Claim 21

The invention, as recited by claim 21, comprises means for inputting a user request at the client terminal (specification page 9, lines 30-33), means for communicating (specification page 10, lines 1-3; Figure 3, "N") between the client terminal and a server part (specification page 10, lines 6-8; Figure 2, "Server"; Figure 4, "Server"), processing means for generating a response from the server part to be transmitted to the client terminal (specification page 9, lines 24-26; Figure 3, "Server Response (Instructions + Data)"), and means for displaying the response at the client terminal (specification page 9, lines 30-33; Figure 2, element 5; Figure 3, "Display Means").

Claim 21 recites the response from the server part comprises instruction data and display data to be displayed (specification page 10, lines 6-8; Figure 3, "Server Response (Instructions + Data)").

Claim 21 also recites that the graphic interface device further comprises i) an association engine at the client terminal to create a visualization model through the association of construction elements based on the instruction data interpreted by the instructions manager (specification page 10, lines 15-22; Figure 3; "Association Engine"), ii) storing means at the client terminal for storing the construction elements at the client terminal (specification page 12, lines 6-10; Figure 3, "Data

Store"; Figure 4, "Cache"), iii) means for merging the visualization model with the display data of the response, a result of the merging to be displayed at the client terminal (specification page 11, lines 12-19; specification page 8, lines 11-23; Figure 3, "Merging Means"), and iv) a rules engine, at the client terminal, configured to provide event-operated interface controls in the visualization model, and script code to manage the event-operated interface controls at the level of the client terminal (specification page 11, lines 20-29; Figure 3, "Rules Engine").

Claim 21 further recites that the construction elements are retrieved from storage at the client terminal and include a descriptive interface of visualization model objects, a presentation layer, and logical rules, the logical rules to be applied at the client terminal to the visualization model (specification page 10, lines 17-23; Figure 3, "Descriptive Interface", "Presentation Layer", "Logical Rules"; Figure 5, "Descriptive Interface", "Presentation Layer", "Logical Rules").

Independent Claim 31

The invention, as recited by claim 31, comprises a client terminal including an input device, a storage device, and a graphical display device to interact with a user (specification page 9, lines 30-33; Figure 2, element 5; Figure 3, "Display Means", "Data Store"; Figure 4, "Cache"), the client terminal in communication with the server part by a communication means (specification page 10, lines 1-3; Figure 3, "N"; Figure 4 generally).

Claim 31 recites that the client terminal includes i) an instructions manager configured to receive a response from a computer server part (specification page 10, lines 10-12; Figure 3, "Instruction Manager"), the response generated as a result of a user request transmitted from the client terminal to the server part (specification page 10, lines 6-8), the response including instruction data and display data (specification page 10, lines 6-8; Figure 3, "Server Response (Instructions + Data)"),

ii) an association engine configured to create a visualization model from a plurality of construction elements, the construction elements retrieved from the storage device, based on the instruction data interpreted by the instructions manager, the construction elements including static model graphic objects, a presentation layer including different elements and attributes related to a graphic presentation, and logical rules (specification page 10, lines 15-22; Figure 3; “Association Engine”),

iii) a storage engine configured to store the construction element at the client terminal (specification page 12, lines 3-10; Figure 3, “Data Store”; Figure 4, “Cache”)),

iv) a merging means configured to merge the visualization model with the display data of the response as a merged result (specification page 11, lines 12-15; Figure 3, “Merging Means”),

v) a rules engine configured to provide event-driven interface controls to the visualization model (specification page 11, lines 20-25; specification page 9, lines 3-6; Figure 3, “Rules Engine”), and Javascript code to manage the event-operated interface controls, based on the logical rules (specification page 9, lines 7-10), and

vi) a transformation engine configured to transform the merged result into display elements for display on a navigator operating on the graphical display device (specification page 11, lines 16-19; Figure 3, “Transformation Means”).

(vi) Ground of Rejection to be Reviewed on Appeal

The ground of rejection on appeal is as follows:

a) whether claims 13, 15-21, and 24-31 were properly rejected as unpatentable under §103(a) as obvious over Davis (U.S. Publication 2002/0130900; hereinafter DAVIS) in view of Sanderson (WIPO Publication WO/2002/044897; hereinafter SANDERSON).

(vii) Arguments

Arguments Concerning Claims 13, 21 and 31, and dependent claims 15-20, 24-30

Independent claims 13, 21, and 31, and dependent claims 17-20, and 24-30 stand together. Dependent claims 15 and 16 are argued separately.

Claim 13

Claim 13 was rejected under §103(a) as obvious over DAVIS in view of SANDERSON.

The Examiner improperly rejects claim 13 because the combination of DAVIS with SANDERSON fails to teach all the elements recited in independent claim 13 of the application, and because there exists no motivation to one of skill in the art to combine SANDERSON with DAVIS as proposed by the Examiner.

Before detailing the Examiner's obviousness rejection, it is useful to review DAVIS and SANDERSON in more detail.

DAVIS teaches an improvement on remote desktop applications, wherein a desktop application running on a server computer may be displayed and controlled remotely on a client computer, similarly to products like Citrix WinFrame® and Microsoft® Terminal Server® (DAVIS, paragraph [0005], lines 1-8). In the latter applications, the remote client receives from the server the entire bitmapped graphical representation of the server-based application, which must be updated in real-time responsive to user interactive events generated by the client (DAVIS, paragraph [0005], lines 5-7). DAVIS improves upon this wherein the client only receives specifications of interface components, encoded in a language like XML, which are rendered on the remote client with the client's native graphical widget set (DAVIS, paragraph [0006]-[0010]). As a result, the server-based application takes on an appearance corresponding to the native interface of the client, and also provides better performance and better compatibility across multiple platforms (DAVIS, column [0006], lines 1-7).

Whereas DAVIS offers an improved system for remotely controlling a graphics-based application running on a server, SANDERSON is directed to an improvement over client-based application platforms like Java applets (SANDERSON, page 2, lines 8-17; page 3 line 19 to page 4 line 4). SANDERSON describes generating a complex user interface on a client by means of receiving and interpreting configuration data and workload data with a declarative user interface (UI) generator (SANDERSON, page 4, lines 5-10). The UI generator may be manifested either as a standalone application on the client or a plug-in to a standard web browser (SANDERSON, page 8, lines 15-19). The UI generator generates a graphical UI on the client to interact with data from one or more server-based, network-deployed applications by first receiving contextual information which defines the type of data to be displayed at the client and the tasks with which the data can be accessed and manipulated (SANDERSON, page 8, lines 3-11). The context information directs the UI generator to select and position user-interface elements to assemble an interface on the client through which the user can access and manipulate server-based data from one or more network-based sources (SANDERSON, page 8, lines 8-13; page 8 line 22 to page 9 line 8).

SANDERSON differs from DAVIS in that SANDERSON teaches a client that generates, at run-time, a graphical UI suitable for one or more server-based applications from no more than the contextual information (SANDERSON, page 8 line 22 to page 9 line 8) while DAVIS teaches a thin-client viewer that receives display information from and transmits user events to an application that is in all other respects operating on the server (DAVIS paragraphs [0008]-[0009]; [0011], lines 1-5). In other words, SANDERSON describes a framework wherein the client generates an graphical UI capable of communicating with several server applications from a small amount of contextual information, and DAVIS teaches a client-based "viewer" (DAVIS, [0007], lines 6-7) to render the graphical UI for an application running remotely on a server, except that the DAVIS "viewer" employs its native UI facilities in place of receiving bitmapped screen shots of the server-based application transmitted from the server (DAVIS, [0007], lines 13-15).

I THE VISUALIZATION MODEL CREATED AT THE CLIENT

The Examiner, in the Official Action of June 20, 2008, offers DAVIS as disclosing a method for graphics-based interfacing between a user and a computer wherein, at the client terminal, a visualization model is created by the association engine according to the interpreted instruction data through the association of construction elements retrieved from storage at the client terminal, the construction elements including a descriptive interface of visualization model objects, a presentation layer, and logical rules.

The Official Action, on page 8, bottom paragraph, acknowledges that DAVIS does not disclose the claimed aspect that display data of the response are merged, at the client terminal, with the visualization model and displayed as a merged result.

The Examiner offers SANDERSON as disclosing the claimed aspect of merging the data aspect at the client terminal, with the visualization model in order to display and displayed as a merged result, wherein the UI generator communicates with databases and initiates a user interface. The Examiner further states that SANDERSON illustrates the aspect of displayed and merged data on the interface in FIG. 1, at 104A.

The Examiner concludes it would have been obvious to one of ordinary skill in the art at the time of the invention to enhance DAVIS's invention with SANDERSON's invention to merge the interface with the data, because all applications need data in order to perform a function. Appellants respectfully disagree.

I.(A) THE REFERENCES DO NOT TEACH THE VISUALIZATION MODEL

Neither DAVIS nor SANDERSON teach or suggest creating a visualization model at the client as recited by claim 13. In particular, neither DAVIS nor SANDERSON teach or suggest the construction elements including a descriptive interface of the visualization model objects, a presentation layer, and logical rules to be applied locally to the visualization model, as required by claim 13.

The Examiner states on page 8 that instruction data are interpreted to construct a visualization model at DAVIS' client terminal. The Examiner offers Figure 3 of DAVIS, stating that viewer 200 renders GUI with native widget on Client computer 202, 316). On page 2 of the Official Action, the Examiner further states that the construction elements of the claim read on the widgets taught by DAVIS.

On the contrary, the graphical UI of DAVIS is constructed at the server prior to its being transmitted to the "viewer" on the client. "The projector 100 component of [DAVIS] is a software application server that runs on a server computer 102... The software developer creates a software application's graphical user interface... by visually laying out the application's windows and widgets (menus, buttons, scroll bars, etc.), and then tying handler code to widgets' events... Upon completion of a software application, the programmer 'publishes' the application to the projector 100 to make the application accessible to the client computer 202," (DAVIS, paragraphs [0019]-[0022]).

Here, the "viewer" at the client of DAVIS does not in fact construct the model. DAVIS' "viewer" merely displays the model using client-based graphical UI elements or "widgets" (paragraph [0030]), the model already having been <u>completely constructed</u> at the server (paragraph [0008], lines 1-7). As stated above, DAVIS' use of client-based widgets is an improvement over the prior art where the server must communicate actual bitmapped picture data of widgets as they would appear on the server. DAVIS' employment of client-based widgets reduces the amount of network traffic necessary to remotely access the server-based application, while simultaneously presenting the remote application in a manner better integrated with the client user's local desktop.

The use of client-based widgets, however, does not make the client any less than completely dependent on the server. "The viewer transmits <u>all</u> user interaction with the graphical user interface to the projector," (paragraph [0011], lines 3-5). DAVIS teaches a <u>thin-client</u> application wherein "[a]ll code execution takes place inside a virtual machine 106 <u>within the projector</u>," (paragraph [0025], lines 3-5). "[T]he viewer 200 is responsible for transmitting all user interaction with the GUI," (paragraph [0031], lines 1-2; emphasis added).

Thus, DAVIS does not teach or suggest creating a visualization model by an association engine at the client terminal, as required by claim 13.

I.(B) THE REFERENCES DO NOT TEACH AN ASSOCIATION ENGINE

Furthermore, neither DAVIS nor SANDERSON teach or suggest the association engine to create the visualization model as recited by claim 13. For example, DAVIS fails to teach the rules engine, as required by claim 13, to apply logical rules at the client as one of the construction elements of the visualization model. The logical rules, as recited by the claim, provide event-operated interface controls in the visualization model and script code to manage the event-operated interface controls at the client terminal.

DAVIS' thin-client model, wherein "the viewer 200 reports to the projector 100 any user interface events," (paragraph [0025], lines 5-8), is in stark contrast to event-operated interface controls managed at the level of the client, as recited. The logical rules of the instant invention enable a local processing at the client terminal in response to user-triggered events generated by the event-operated interface controls. The thin-client model of DAVIS does not teach this.

For example, DAVIS does not teach script code to manage event-operated interface controls with the visualization model, as required by claim 13. On the contrary, DAVIS describes a system wherein an XML stream includes information about widgets required by the published application, so that the client computer can provide compatible widgets from its native interface toolkit (paragraph [0030]). All user interaction with the graphical UI is transmitted from the "viewer" to the server, (paragraph [0031], lines 1-2). Whenever the user initiates an event while accessing an application and clicking on a button or pressing "enter," for example, the viewer sends the event to the server "projector" to handle the event and direct the viewer how to respond (paragraph [0031], lines 2-6), and the server directs the "viewer" how to respond (paragraph [0031], lines 6-7). There is no disclosure in DAVIS wherein the viewer at the client executes scripts.

Moreover, DAVIS expressly teaches away from the use of scripts and/or a script-compatible web browser as the client "viewer" on the end-user terminal. "[It] takes less time for developers to create software applications in accordance with the present invention [of DAVIS] because they do not have to bother with entering data into web pages in HTML, JavaScript, VBScript, ASP, Java, etc." (paragraph [0032], lines 8-12).

Similarly, SANDERSON teaches away from the use of script languages, disclosing "the declarative UI generator does not require the extensive programming typically associated with scripts and applets," (page 3, lines 23-24). SANDERSON discloses the use of a web browser by way of either a plug-in or a network distributable applet (page 8, lines 17-19), but also teaches an alternative embodiment where no web browser is required. While SANDERSON teaches an embodiment where the UI generator is a Java applet running within a browser (page 12 line 21 to page 13 line 6), it is noted that one of skill readily understands that Java, requiring compilation into machine code or byte-code to execute within a Java Virtual Machine, is not a scripting language to manage UI events in a browser, e.g. JavaScript.

On page 3 of the Official Action, the Examiner argues that DAVIS offers paragraph [0032] of the reference as disclosing that JavaScript is well known in the art at the time of the invention. However, the passage referred to by the Examiner teaches that DAVIS "does not require a web browser", (DAVIS paragraph [0032], lines 3-4), wherein the viewer "runs directly and locally on the client computer 202 to provide the user with an easy-to-use, familiar environment," (paragraph [0032], lines 4-6) which "takes less time for developers to create software applications... because they do not have to bother with entering data into web pages in HTML, JavaScript, VBScript, ASP, Java, etc.," (paragraph [0032], lines 8-12). Nowhere in this passage or in any other passage of DAVIS does the reference teach or suggest logical rules applied at the client terminal by a rules engine providing event-operated interface controls in the visualization model and script code to manage the event-operated interface controls, as recited by claim 13.

I. (C) THE REFERENCES DO NOT TEACH LOGICAL RULES

The Examiner further argues at the bottom of page 9 of the Official Action that SANDERSON teaches the logical rules recited by claim 13. The Examiner offers Figure 2 of SANDERSON, stating that a validator 211 and format 212 are illustrated which apply logical rules, and that validation is based on certain criteria.

Appellants respectfully disagree. SANDERSON teaches only that "a validator object 211 is an object which can validate the value encapsulated by specified data element 208," (page 19, lines 14-16). SANDERSON makes no teaching or suggestion that the validator object teaches a rules engine providing event-operated interface controls in the visualization model and script code to manage the event-operated interface controls, as recited by claim 13. On the contrary, no other teaching or detail is disclosed by SANDERSON of the validator object anywhere else in the reference.

Therefore, it is respectfully submitted that the recitation of a visualization model created as a merged result by an association engine as recited by claim 13 is novel and non-obvious over DAVIS in view of SANDERSON.

II. THE REFERENCES DO NOT TEACH THE RESPONSE AND THE DISPLAY DATA OF THE RESPONSE MERGED WITH THE VISUALIZATION MODEL

It is further respectfully submitted that neither DAVIS nor SANDERSON teach or suggest that the response from the server comprises instruction data and display data, and that the display data of the response is merged with the visualization model at the client terminal and displayed at the client terminal, as required by claim 13.

On page 8 of the Office Action, last paragraph, the Examiner acknowledges that DAVIS does not specifically teach merging data at the client terminal as recited by the Appellants' claim. The Examiner offers page 9, paragraph [0010], lines 1-3 of SANDERSON, stating that SANDERSON discloses merging data, at the client terminal, with the visualization model in order to display a merged result, wherein a UI generator communicates with databases and initiates a user interface (paragraph spanning pages 8-9 of the Official Action). The Examiner further states that SANDERSON specifically illustrates the aspect of displayed and merged data on the interface in Figure 1 at element 104A.

The Examiner concludes that one of skill would be motivated to combine this disclosure of SANDERSON to enhance DAVIS in order to merge the interface with data because "all applications need data in order to perform a function," (page 9 of Official Action, lines 3-6).

Appellants respectfully disagree. No combination of DAVIS and SANDERSON teaches or suggests a method for graphic interfacing between a user and a computer system wherein the server response comprises instruction data and display data, and display data of the response are merged with the visualization model and displayed as a merged result at the client terminal, as recited by claim 13 (emphasis added). It is also respectfully submitted that the rationale offered by the Examiner is improper for being conclusory.

SANDERSON teaches, at page 9 lines 6-8 (i.e., lines 1-3 of the paragraph to the right of the marker "10" in the left-hand margin of page 9), "in accordance with the present invention, the declarative UI generator can communicate with disparate server-side applications, application servers, and databases and further can initiate user interaction." This passage makes no teaching of the response comprising display data combined with the visualization model as recited in the claim.

Further, Figure 1, element 104A of SANDERSON illustrates contextual information 107A and content 107B stored and retrieved from a fixed storage 108 in the content server 106 (page 11, lines 4-6 and lines 18-20). SANDERSON neither illustrates nor describes contextual information 107A and content 107B transmitted together as a response to the client. At best, Figure 1 teaches only that contextual information 107A and content 107B are both stored on the fixed storage 108 "in the content server 106," (page 11, line 6).

SANDERSON teaches a context element, 107A, and a content object, 107B stored and retrieved from the server (page 11, lines 7-25). SANDERSON discloses context 107A as contextual information relating to data to be displayed by the UI generator 103 (page 11, lines 7-8), while content 107B is described as an abstraction that allows various representations of data and data streams to be communicated between the generator 103 and the content server 106 (page 12, lines 2-4).

SANDERSON teaches "the configuration and specification data can establish <u>how</u> the declarative UI generator 103," at the client terminal, "can <u>request</u> content 107B from the content server 106, for example by way of an object request broker," (page 11, lines 18-20; emphasis added). That is, the data configuration and specification data 107A instructs the declarative UI generator on the client side how to interface with the content server 106 <u>to receive</u> the content 107B, and therefore would not receive the data configuration and specification data 107A and the content 107B <u>together</u> in a server response as required by the claim.

SANDERSON thus teaches receiving content 107B separately from specification data 107A, the latter specification data 107A instructing SANDERSON's client-side application how to subsequently retrieve the content 107B after the specification data 107A has been received, interpreted, and executed by the client. DAVIS, by comparison, receives all visualization data and content data simultaneously, and <u>already merged</u>, from the server, the DAVIS client doing little more than invoking local widget routines to render, or draw, the various UI elements, where the prior art to DAVIS would receive bitmapped graphics data from the server for display on the client (DAVIS, paragraphs [0006]-[0010]). That is, SANDERSON and DAVIS take <u>opposite approaches</u> to the delivery of visualization and content data and merger of these distinct data streams into a result for display, SANDERSON defining the visualization at the client and merging data delivered separately from the server, and DAVIS defining the visualization and merging the data with the visualization at the server before transmitting the result to the client.

Thus DAVIS discloses a thin client requiring higher bandwidth, while SANDERSON teaches a general-purpose UI generator that cannot be thin due to the complexity of tasks required by its declarative UI generator to connect to servers and retrieve context data based on pre-loaded specification data.

In contrast, the present invention is more balanced and does not suffer from the limitations of high bandwidth or client resources of either of DAVIS or SANDERSON, because the client receives instruction data and display data together from the server, fully describing the visualization model but constructed and merged with data <u>locally at the client</u>. Neither SANDERSON nor DAVIS, individually or in combination, offer any teaching that either describes or suggests the arrangement and steps as recited in claim 13.

The Examiner further fails to offer a rational motivation to one of skill to consider a combination of the two, opposite teachings of the references. On the contrary, the Examiner offers only that one of skill would "merge interface with data, because all applications need data in order to perform a function," (Official Action page 9, lines 3-6). The Examiner's statement is conclusory because each of SANDERSON and DAVIS <u>already</u> teach merging interface with data, each in a manner distinct from both each other and the present invention as claimed. The Examiner offers no reference or rationale to overcome these teachings of the references, each of which teach away from the other with respect to receiving and merging visualization data. Thus, there is no motivation to one of skill to combine SANDERSON with DAVIS.

Accordingly, it is respectfully submitted that neither DAVIS nor SANDERSON, individually or in combination, teach or suggest the response comprising instruction data and display data to be displayed, and that display data of the response are merged, at the client terminal, with the visualization model and displayed as a merged result, as recited by claim 13. Therefore, it is respectfully submitted that this recitation is novel and non-obvious over DAVIS in view of SANDERSON.

For all the foregoing, it is respectfully submitted that the rejection of claim 13 is in error, and that claim 13, 21, and 31, and claims depending therefrom, are patentable over DAVIS in view of SANDERSON. Reversal of the Examiner's rejection is earnestly requested.

Claim 15

Claim 15 stands alone. Claim 15 was rejected as obvious over DAVIS in view of SANDERSON.

The Examiner improperly rejects claim 15 because the combination of DAVIS with SANDERSON fails to teach all the elements recited in claim 15 of the application.

The Examiner, in the first paragraph of page 10 of the Official Action, states that SANDERSON, achieves the claimed aspect of language resources locally available or downloadable from the server part, one is associated to the created visualization model, wherein in Fig. 1 the declarative User Interface is displayed in that associated language. The Official Action does not identify with specificity the element in DAVIS or SANDERSON purportedly corresponding to the language resources.

Appellants respectfully disagree. It is respectfully submitted that neither DAVIS nor SANDERSON, individually or in combination, describe or suggest associating a language resource from a plurality of language resources, nor providing a designation of the predetermined language with the instruction data provided by the server part, as required by claim 15.

On the contrary, neither DAVIS nor SANDERSON describe or suggest any mechanism or resource related to a plurality of language resources, stored at the client terminal or elsewhere.

In the paragraph spanning page 3 and 4 of the Official Action, the Examiner states that SANDERSON discloses in Figure 4B a context file, 426, 450, and at step 428, 432 context file is parses and the content specifications are extracted, wherein a new set of content specifications and task descriptions can be extracted, referencing page 26, lines 5-10 of the specification. The Examiner proceeds to conclude that one of skill would have found it obvious to "have the language specified in the context file, because this would allow different language usages," (Official Action page 4, liens 5-6.

It is respectfully submitted that this statement is improper for being conclusory, as the Examiner has failed to provide any teaching, in either of DAVIS or SANDERSON, of the recited feature.

On the contrary, the context file taught by SANDERSON includes structure and semantics of data to be communicated between the declarative UI generator 103 and the content server 106, so that the declarative UI generator 103 can identify the <u>types</u> of data elements to be displayed in the generator, the interrelationship between each data element, and the types of tasks to be performed with the data elements through the generator (SANDERSON, page 12, lines 6-15). Accordingly, "suitable UI elements 104A, 104B can be selected, initialized and displayed in the UI generator 103 such that the UI elements 104A, 104B are consonant with the <u>types</u> of data elements to be displayed," (page 12, lines 16-19).

There is no teaching in either DAVIS or SANDERSON of any teaching or suggestion of a <u>plurality of language resources</u> as recited by claim 15. The context file is described by SANDERSON only toward data types and elements, not toward language or a plurality of language resources.

Therefore, it is respectfully submitted that claim 15 is novel and non-obvious over DAVIS in view of SANDERSON. Reversal of the Examiner's rejection of claim 15 is earnestly requested.

Claim 16

Claim 16 stands alone. Claim 16 was rejected as obvious over DAVIS in view of SANDERSON.

The Examiner improperly rejects claim 16 because the combination of DAVIS with SANDERSON fails to teach all the elements recited in claim 15 of the application.

The Examiner, in the second paragraph of page 10 of the Official Action, states that DAVIS achieves the aspect of some personalization display filters are associated to the visualization model in order to modify the visual rendering of the default visualization model at the level of the client terminal, wherein projector 100 transmits XML stream of interlace component of software application to viewer 200 and viewer 200 has information about what each widget needed.

The Examiner further states that SANDERSON discloses the claimed aspect of some personalization display filters wherein the configuration data and context file is parsed in Fig. 4B, step 428 to obtain the workflow description and content specification.

Appellants respectfully disagree. It is respectfully submitted that neither DAVIS nor SANDERSON, individually or in combination, describe or suggest associating personalization display filters at the client terminal to the visualization model in order to modify a visual rendering of the visualization model according to specific client parameters, as required by claim 16.

On the contrary, DAVIS provides no teaching or suggestion of either specific client parameters or a step of associating personalization display filters to a visualization model to <u>modify</u> the visual rendering of the visualization model. That is, the visualization model is created by the construction elements, as recited in claim 13. Claim 16 recites the <u>modification</u> of the visual rendering of the visualization model so created, according to client parameters not recited in the creation of the visualization model. Davis does not teach this additional step of modifying the visual model already created by the elements in claim 13.

Furthermore, SANDERSON teaches that developers, particularly web site developers, need only provide a content specification to the declarative UI generator 103 specifying the type of data to be displayed in the dynamically generated UI and the tasks with which the data can be accessed and manipulated (page 13, lines 18-24). SANDERSON provides no teaching or suggestion that a user, as opposed to a developer, can modify a visual appearance of the dynamically generated UI.

SANDERSON teaches that personal information, such as credit card data, may be stored on the client-side file system (page 13, lines 2-5). SANDERSON does not, however, teach or suggest, however, any preferences modifying the visual rendering of the visualization model, as required by claim 16.

The Examiner, in the first full paragraph on page 4 of the Official Action, states that DAVIS discloses some personalization display filters are associated to the visualization model in order to modify the visual rendering of the default visualization model at the level of the client terminal, wherein projector 100 transmits the XML stream of interface components for the server-based application to the client-based viewer 200, and viewer 200 has information about each locally-based widget needed. The Examiner states that DAVIS discloses that the interface component includes information about all the widgets needed to execute the application, such as their placement size, and captions, and the viewer 200 renders the graphical user interface with the native widget set of the client computer's operating system, based on the widget information of the projector. The Examiner concludes that the native widget set dictates the appearance of the widgets, e.g., their style and shape, and the interface for the application looks and feels like a native desktop application (DAVIS, Abstract).

The Examiner, however, has failed to establish that DAVIS teaches personalization display filters that <u>modify</u> the visual rendering of the visualization model. DAVIS does not teach this.

On the contrary, DAVIS teaches only that the appearance of the published application in the client-based viewer will coincide with the appearance of the local widgets (paragraph

[0030], lines 24-28). “The application’s windows, buttons, text boxes, menus, etc. look and feel just like those of other application that are run on the client desktop, no matter what desktop operating system is running on the client computer 202,” (paragraph [0035], lines 7-10). DAVIS provides no teaching or suggestion of any step that would further modify the rendering of the appearance of the widgets by the “viewer” on the client. There is no disclosure, however, of additional factor that would <u>further</u> influence, or <u>modify</u> the appearance of the widgets, e.g., color, placement, style, or size, based on a <u>personal</u> preference or parameter, i.e., personalization.

In other words, there is no teaching or suggestion of any step or mechanism that would cause the appearance of the widgets of an application rendered on a particular terminal to ever have any but one single appearance. It is therefore respectfully submitted that DAVIS does not teach a personalization display filters as recited by claim 16.

The Examiner further states that SANDERSON teaches associating personalization display filters at the client terminal to the visualization model as recited in claim 16 in Figure 4B at step 428, to obtain the workflow description and content specification, stating that SANDERSON discloses that “the user will provide a content specification to the declarative User Interface generator 103 specifying the type of data to be displayed in the dynamically generated UI and the tasks with which the data can be accessed and manipulated through the dynamically generated UI (Official Action page 5, lines 1-5. The Examiner references SANDERSON, page 13, the paragraph by the line-marker 20.

This is not correct. SANDERSON, at page 13, lines 19-24, discloses "<u>Web site developers</u> need only provide a content specification to the declarative UI generator 103 specifying the type of data to be displayed in the dynamically generated UI and the tasks with which the data can be accessed and manipulated through the dynamically generated UI," (emphasis added). There is not teaching or suggestion of <u>users</u> having any impact on the type of data displayed, nor is there any teaching in this passage or any other passage of SANDERSON teaching or suggesting personalization display filters as recited by claim 16.

At best, SANDERSON discloses, on page 13, lines 2-6, that the declarative UI generator 103 can interact with a client-side file system to store preferences or personal information, such as credit card data, and can also concurrently connect to multiple servers, particularly web sites. This, disclosure, however, makes no teaching as to the <u>modification of the visual rendering of the visualization model</u>, as required by claim 16. As with DAVIS, above, SANDERSON makes no teaching or suggestion of a variation of its visual appearance outside of what is predetermined based on the client widget set and the instructions provided by the application server

Therefore, it is respectfully submitted that claim 16 is not rendered obvious under DAVIS in view of SANDERSON. Reversal of the Examiner's rejection of claim 16 is earnestly requested.

Conclusion

For all the foregoing, the obviousness rejection is improper. Reversal of the obviousness rejection is therefore earnestly requested.

Please charge the requisite Appeal Brief fee of $510 to our credit card.

Respectfully submitted,

YOUNG & THOMPSON

By /Roland E. Long, Jr./

Roland E. Long, Jr.
Attorney for Appellants
Registration No. 41,949
209 Madison Avenue, Suite 500
Alexandria, VA 22314
Telephone: 703/521-2297

REL/mjr
August 20, 2008

(viii) Claims Appendix

1-12 (cancelled)

13. Method for graphic interfacing between a user and a computer system in which the following operations are performed:

- inputting a user request at a client terminal,

- transmitting the user request from the client terminal to a server part for processing by the server part,

- receiving a response to the request at the client terminal, the response generated by the server part,

- displaying the response at the client terminal,

wherein:

- the response comprises instruction data and display data to be displayed;

- at the client terminal, the instruction data are interpreted and forwarded to an association engine;

- at the client terminal, a visualization model is created by the association engine according to the interpreted instruction data through the association of construction elements retrieved from storage at the client terminal, the construction elements including a descriptive interface of visualization model objects, a presentation layer, and logical rules;

- display data of the response are merged, at the client terminal, with the visualization model and displayed as a merged result; and

- the logical rules applied at the client terminal to the visualization model by a rules engine, providing event-operated interface controls in the visualization model and script code to manage the event-operated interface controls at the level of the client terminal.

14. (cancelled)

15. Method according to the claim 13, further comprising the step of:

at the client terminal, associating a language resource from a plurality of language resources to the visualization model, the plurality of language resources stored at the client terminal, to adapt the visualization model to a predetermined language, a designation of the predetermined language, to be associated to the visualization model, provided in the instruction data from the server part.

16. Method according to claim 13, further comprising the step of:

at the level of the client terminal, associating personalization display filters to the visualization model in order to modify a visual rendering of the visualization model according to specific client parameters.

17. Method according to claim 13, wherein

the instruction data include an indication of a type of construction elements characterizing the visualization model to be created.

18. Method according to claim 13, wherein

data resident at the client terminal are updated at the client terminal through the following steps:

- at the server part, generating a storing message which includes storing instruction data and data to be stored,

- transmitting the storing message from the server part to the client terminal,

- at the client terminal, storing the data to be stored in a memory device local to the client terminal in a manner according to the storing instruction data.

19. Method according to claim 13, wherein display is performed at the client terminal through the use of a navigator.

20. Method according to claim 13, wherein,

at least a portion of the data to be displayed and at least a portion of the construction elements use a XML format, and the merged result is translated to HTML format in order to be displayed.

21. Graphic interface device between a user and a computer system, comprising:

• means for inputting a user request at the client terminal;

• means for communicating between the client terminal and a server part; and

• processing means for generating a response from the server part to be transmitted to the client terminal; and

• means for displaying the response at the client terminal,

wherein,

- the response from the server part comprises instruction data and display data to be displayed,

- said graphic interface device further comprising: an instructions manager at the client terminal to interpret the instruction data received from the server part,

- said graphic interface device further comprising: an association engine at the client terminal to create a visualization model through the association of construction elements based on the instruction data interpreted by the instructions manager, the construction elements retrieved from storage at the client terminal and including a descriptive interface of visualization model objects, a presentation layer, and logical rules, the logical rules to be applied at the client terminal to the visualization model;

- said graphic interface device further comprising: storing means at the client terminal for storing the construction elements at the client terminal;

- said graphic interface device further comprising: means for merging the visualization model with the display data of the response, a result of the merging to be displayed at the client terminal; and

- said graphic interface device further comprising: a rules engine, at the client terminal, configured to provide event-operated interface controls in the visualization model, and script code to manage the event-operated interface controls at the level of the client terminal.

22-23. (cancelled)

24. Device according to claim 21, further comprising:

at the client terminal, a navigator to display the merging result.

25. Method according to the claim 13, wherein the script code is provided in the Javascript scripting language.

26. Method according to the claim 13, wherein the visualization model comprises images, script code, and markups, the script code being in the Javascript scripting language, the markups being in Hypertext Markup Language (HTML).

27. Device according to claim 21, wherein the script code is provided in the Javascript scripting language.

28. Device according to claim 21, wherein the visualization model comprises images, script code, and markups, the script code being in the Javascript scripting language, the markups being in Hypertext Markup Language (HTML).

29. Device according to claim 21, further comprising:

a plurality of language resources stored at the client terminal, the instruction data providing a designation of one of the plurality of language resources to be associated with the visualization model, the association model adapting the visualization model to the designated language resource.

30. Device according to claim 21, further comprising:

personalization display filters configured to modify a visual rendering of the visualization model according to specific client parameters.

31. A graphic interface device between a user and computer system, comprising:

a client terminal including an input device, a storage device, and a graphical display device to interact with a user, the client terminal in communication with the server part by a communication means, wherein,

the client terminal includes

i) an instructions manager configured to receive a response from a computer server part, the response generated as a result of a user request transmitted from the client terminal to the server part, the response including instruction data and display data,

ii) an association engine configured to create a visualization model from a plurality of construction elements, the construction elements retrieved from the storage device, based on the instruction data interpreted by the instructions manager, the construction elements including static model graphic objects, a presentation layer including different elements and attributes related to a graphic presentation, and logical rules,

iii) a storage engine configured to store the construction element at the client terminal,

iv) a merging means configured to merge the visualization model with the display data of the response as a merged result,

v) a rules engine configured to provide event-driven interface controls to the visualization model, and Javascript code to manage the event-operated interface controls, based on the logical rules, and

vi) a transformation engine configured to transform the merged result into display elements for display on a navigator operating on the graphical display device.

(ix) **Evidence Appendix**

None.

(x) Related Proceedings Appendix

None.